Microsecond delay in STM32

HAL_Delay can provide a minimum 1 ms delay, but when it comes to microseconds, there isn't any predefined function to create 1 us delay in HAL Library. In this tutorial, we will see how to create microsecond delays in STM32. We will be using one of the Timer to do so. The process will be same for all the STM32 devices, you need to make some minor changes though.

I am using STM32F030F4P6 controller with CUBEIDE but as I mentioned, it will work same for other microcontrollers with any other IDE also.

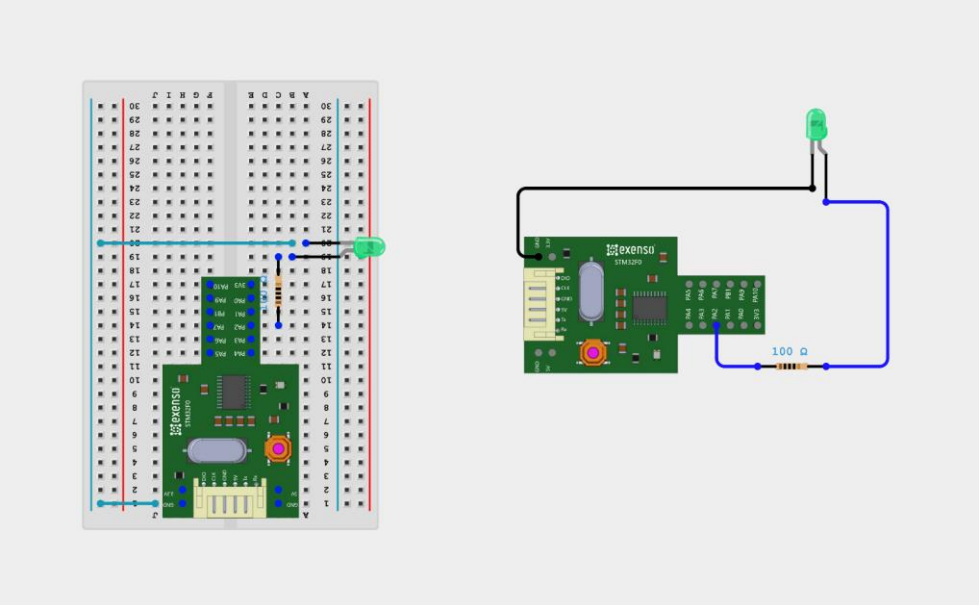
Components Required

You will need the following components –

- 1 × Breadboard
- 1 × STM32F030F4P6
- 1 × LED
- 1 × 100Ω Resistor
- 2 × Jumper

Procedure

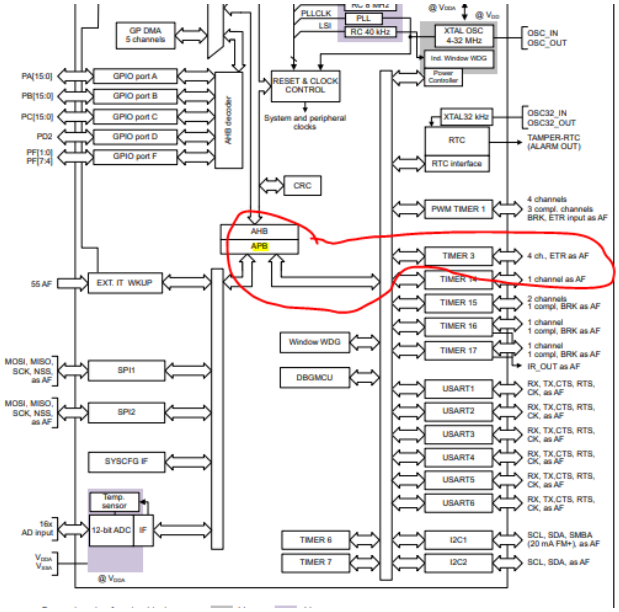
Follow the circuit diagram and hook up the components on the breadboard as shown in the image given below.



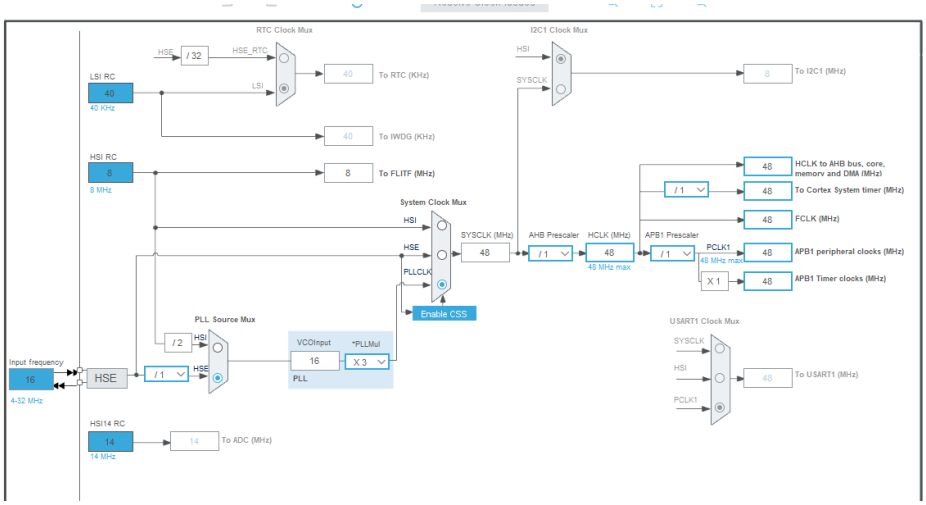
Setup

This is the most important part of this process, so read this section very carefully. First of all, we need to decide which Timer to use for the process. There is no special requirement, just choose anyone. I will use timer 3.

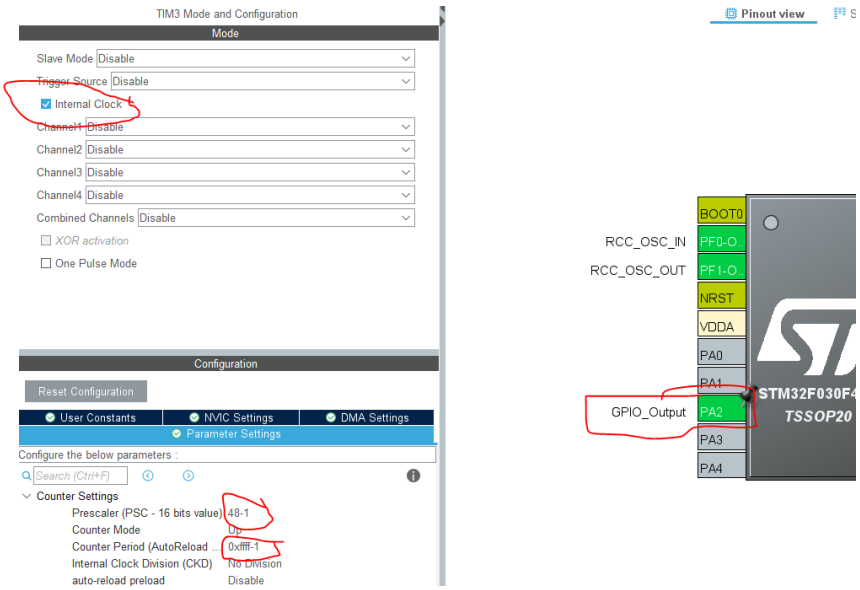
As you can see in the picture below (from the datasheet), the TIM3 is connected to the APB1 clock. This is a very important piece of information, and you should know it about the timer that you are going to use.



Now, let's start the cube IDE, and open the clock setup tab. Here you can see that, once I set the HCLK at MAX i.e. 48MHz, the APB1 clock is also at 48MHz.



This means that the TIMER3 is also running at 48MHz, as the TIM3 is connected to the APB1. Now, let's reduce this frequency in the timer setup section.



- First of all, set the clock source as an internal clock.
- Prescaler divides the Timer clock further, by the value that you input in the prescaler.
- As we want the delay of 1 microsecond, the timer frequency must be (1/(1 us)), i.e 1 MHz. And for this reason, the prescaler value is 48.
- Note that it's 48-1, because the prescaler will add 1 to any value that you input there.
- The ARR I am setting is the max value it can have.
- Basically, the counter is going to count from 0 to this value. Every count will take 1 us. So setting this value as high as possible is the best, because this way you can have large delays also.
- I have set it to 0xffff-1, and it is the maximum value that a 16-bit register (ARR) can have.

I have also enabled the pin PA2 as output, so that we can see the result in an oscilloscope or Logic Analyzer.

Code:

```
#include "main.h"

TIM_HandleTypeDef htim3;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM3_Init(void);

void delay_us (uint16_t us)
{
	__HAL_TIM_SET_COUNTER(&htim3,0);  // set the counter value a 0

	 // wait for the counter to reach the us input in the parameter
	while (__HAL_TIM_GET_COUNTER(&htim3) < us);  }


int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_TIM3_Init();
  HAL_TIM_Base_Start(&htim3);

  while (1)
  {

	  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_2);
	 delay_us(10); // Delay 10 Mircosecond

  }

}
```

Output: